## **Lab Assignment 12**

1. Implement the Observer Pattern for the temperature measurement problem. In this problem, you will have two observers inheriting the <code>ObserverInterface</code> class. One (<code>ConcreteObserverCurrent</code>) is to keep track of the current temperature, and the other (<code>ConcreteObserverStatistics</code>) is to keep track of the max temperature.

Your source of data is from the ParaWeatherData class, which inherits the WeatherDataInterface class. Whenever there is a change for the data, in other words when the sensorDataChange() function is executed, the observers will be notified and the data being stored by the observers will be updated.

The following are predefined interfaces you will include in your program.

```
class ObserverInterface{ //abstract observer
public:
    virtual void update() = 0;
    virtual void show() = 0;
};

class WeatherDataInterface{ //abstract subject
public:
    virtual void registerOb(ObserverInterface* ob) = 0;
    virtual void removeOb(ObserverInterface* ob) = 0;
    virtual void notifyOb() = 0;
};
```

Please complete the classes so that the following main method (you CANNOT change) will show correct results.

```
int main() {
    ParaWeatherData* wdata = new ParaWeatherData;
    ConcreteObserverCurrent* current = new
ConcreteObserverCurrent(*wdata);
    ConcreteObserverStatistics* statistcs = new
ConcreteObserverStatistics(*wdata);

    wdata->sensorDataChange(28.2);
    wdata->sensorDataChange(30.12);
    wdata->sensorDataChange(26);

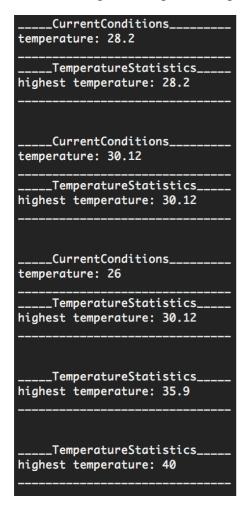
    wdata->removeOb(current);

    wdata->sensorDataChange(35.9);
    wdata->sensorDataChange(40);
```

```
delete statistcs;
delete current;
delete wdata;

return 0;
}
```

The following is the expected output:



**2.** Implement the Command Pattern for the light switching problem. The following are classes you will include in your program:

```
/*the Command interface*/
class Command {
  public:
    virtual void execute()=0;
};

/*Receiver class*/
class Light {
  public:
    Light() {}
    void turnOn() { cout << "The light is on" << endl;}</pre>
```

## **Object-Oriented Programming Language**

06/11/2020

```
void turnOff() { cout << "The light is off" << endl;}
void blink() { cout << "The light is blinking" << endl; }
};</pre>
```

The following is the main method you must NOT change.

```
int main(){
   Light lamp;
   FlipUpCommand switchUp(&lamp);
   FlipDownCommand switchDown(&lamp);
   BlinkCommand blink(&lamp);
   cout << "Switch 1:" << endl;</pre>
   Switch s1;
   s1.setFlipUpCommand(&switchUp);
   s1.setFlipDownCommand(&switchDown);
   s1.flipUp();
   s1.flipDown();
   cout << "Switch 2:" << endl;</pre>
   Switch s2;
   s2.setFlipUpCommand(&switchUp);
   s2.setFlipDownCommand(&switchDown);
   s2.setSpecialCommand(&blink);
   s2.flipUp();
   s2.flipDown();
   s2.special();
   return 0;
```

The following is the expected output:

```
Switch 1:
The light is on
The light is off
Switch 2:
The light is on
The light is off
The light is blinking
```